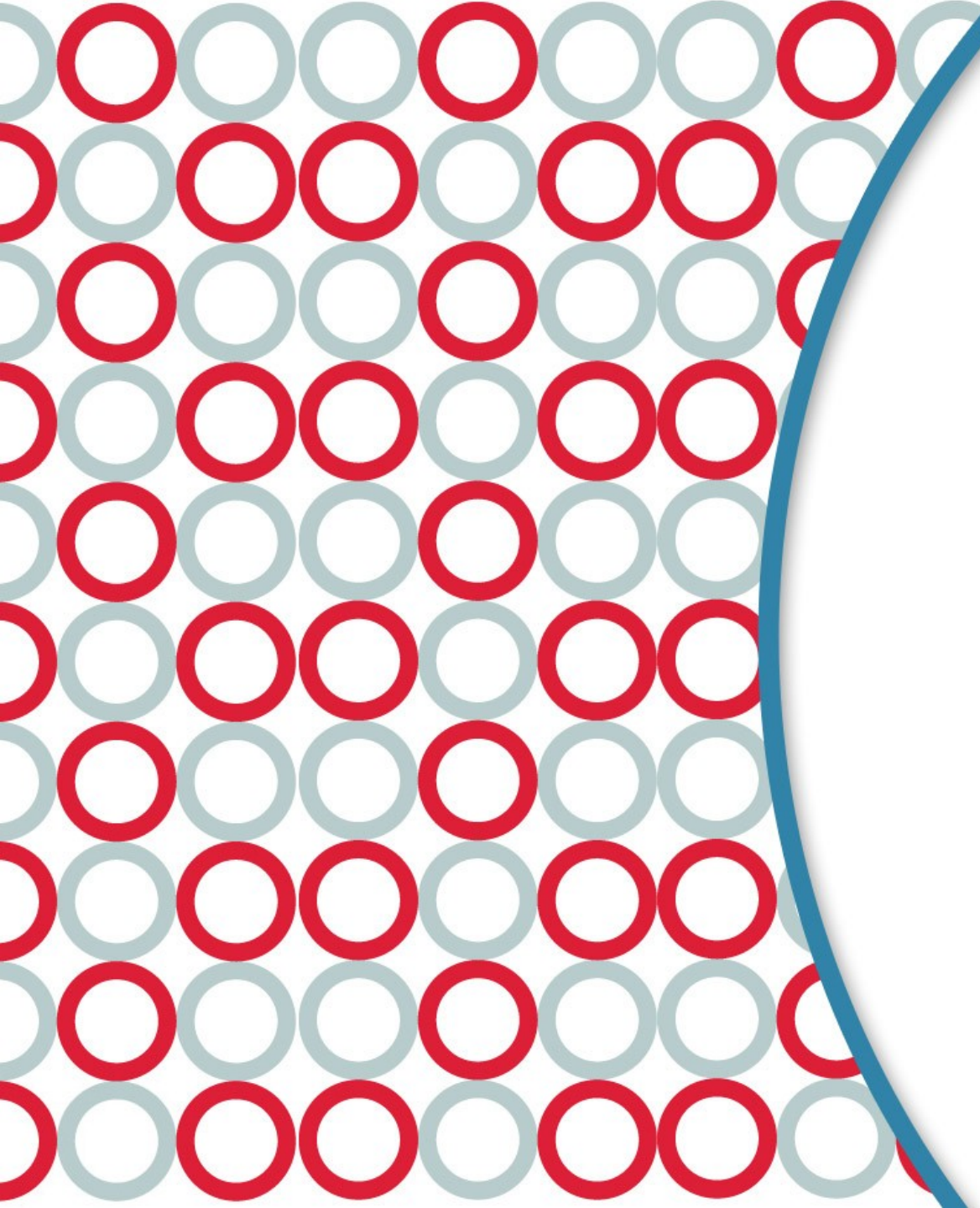


Feb 2020, IPT Course
Java Web Debelopment

Persisting Information with Java

Trayan Iliev
tiliev@iproduct.org
http://iproduct.org

Where to Find the Code?

Intarmediate Java Programming projects and examples are available @ GitHub:

https://github.com/iproduct/course-java-web-development

Agenda

1. Using java Properties class
2. Serialization and deserialization of objects
3. XML Processing and *XML Schema* types,
4. Persistening objects in *XML* using the *Java Architecture for XML Binding (JAXB)*
5. Java to schema and schema to java,
6. Using *schemagen* and *xjc* tools,
7. *JAXB* main annotations, classes, and interfaces
8. M*arshaling* and *unmarshaling* Java objects to/from XML documents, validation.
9. Adding XML based persistence to *MyLibrary* project

Property files, ResourceBundles, I18N and L10N

❖Property files, XML properties:
https://docs.oracle.com/javase/tutorial/essential/environment/properties.html

- MyLabels.properties:

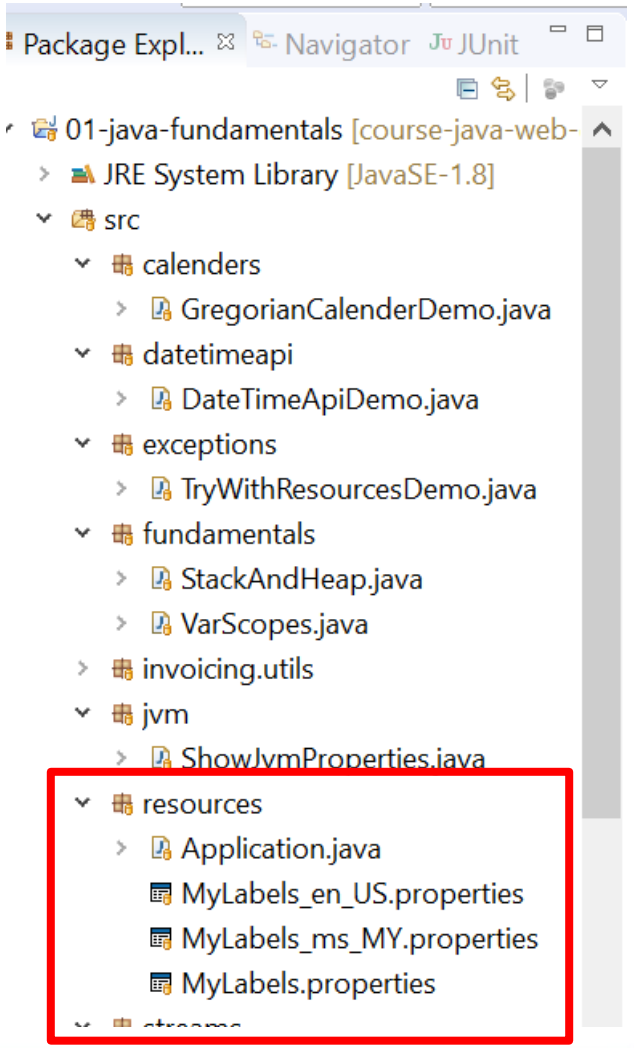
```
how_are_you = How are you?
```

- MyLabels_en_US.properties:

```
how_are_you = How are you?
```

- MyLabels_ms_MY.properties:

```
how_are_you = apa khabar
```



Property files, ResourceBundles, I18N and L10N

- Working with property files, XML properties, classes: https://www.baeldung.com/java-properties

- Example:

```
String rootPath =
Thread.currentThread().getContextClassLoader()
      .getResource("").getPath();
String appConfigPath = rootPath + "app.properties";
Properties appProps = new Properties();
appProps.load(new FileInputStream(appConfigPath));
String appVersion = appProps.getProperty("version");
System.out.println("Version: "+ appVersion);
```

Bean Serialization

- Interface ***Serializable*** – all fields are serialized automatically, except those declsared as ***transient***
- Интерфейс ***Externalizable*** – we serialize everything explicitly
- Implementing private methods ***readObject()*** and ***writeObject()*** – *Serializable* with customization when necessary
- Implementation examples

Programmatic Interfaces for XML Processing

- Simple API for XML (SAX) – event-driven interface using which the document is processed sequentially and its content is reported by the parser by invoking multiple **callback** methods on the processing object (**handler**).
- Document Object Model (DOM) – loads (builds object representation) and manipulates the whole XML document tree in memory.
- Pull Parsing: Streaming API for XML (StAX) – treats the whole document as a sequence of nodes, which are visited sequentially using the **Iterator** desing pattern
- Data Binding (object to XML document): Java Architecture for XML Binding (JAXB) – allows to automate the process of persisting/extracting of data to/from XML (marshalling / unmarshalling)

Java™ API for XML Processing (JAXP)

- Simple API for XML (SAX)
  - Package: `org.xml.sax`
  - Factory class: SAXParserFactory
  - Parser interfaces: SAXParser обвива SAXReader
- Common interfaces to different SAX and DOM parser implementations – package `javax.xml.parsers`

Java API for XML Processing (JAXP) (2)

- Document Object Model (DOM)
  - Package: `org.w3c.dom`
  - Factory class: DocumentBuilderFactory
  - Document model builder class: DocumentBuilder
- Extensible Stylesheet Language Transformations (XSLT)
  - Package: `javax.xml.transform`
  - Factory class: TransformerFactory
  - XML transformer class: Transformer

Simple API for XML (SAX)

- SAX packages:
  - **org.xml.sax** - SAX interfaces
  - **org.xml.sax.ext** - SAX2 extensions
  - **org.xml.sax.helpers** – helper classes for SAX parsers (class DefaultHandler)
  - **javax.xml.parsers** – class SAXParserFactory, which creates and returns SAXParser, Exceptions

Simple API for XML (SAX) (2)

- SAX main classes and interfaces:
  - SAXParserFactory
  - SAXParser
  - SAXReader
  - DefaultHandler
  - ContentHandler
  - ErrorHandler
  - ErrorHandler
  - EntityResolver
  - EntityResolver2
  - LexicalHandler
  - Attributes
  - Attributes2
  - Locator
  - Locator2
  - DeclHandler

Document Object Model (DOM)

- DOM packages:
  - **org.w3c.dom** - DOM API for accessing and manipulation of XML element data, defined by W3C.
  - **javax.xml.parsers** – common classes to access different DOM implmentations: **DocumentBuilderFactory** and **DocumentBuilder** the concrete implmentation is configured using the system property **javax.xml.parsers.DocumentBuilderFactory**, **Exceptions**

Document Object Model (DOM) (2)

- DOM main classes and interfaces:
  - DocumentBuilderFactory
  - DocumentBuilder
  - Attr
  - CDATASection
  - CharacterData
  - Comment
  - Document
  - DocumentFragment
  - DocumentType
  - DOMConfiguration
  - DOMError
  - DOMErrorHandler
  - DOMLocator
  - DOMStringList
  - Element
  - Entity

Document Object Model (DOM) (3)

- DOM main classes and interfaces:
  - EntityReference
  - NamedNodeMap
  - NameList
  - Node
  - NodeList
  - Notation
  - ProcessingInstruction
  - Text
  - TypeInfo
  - UserDataHandler

Extensible Stylesheet Language Transformations (XSLT)

- XSLT packages:
  - **javax.xml.transform** - TransformerFactory и Transformer класове – transform (source, result)
  - **javax.xml.transform.dom** – DOM sources & results
  - **javax.xml.transform.sax** – SAX sources & results
  - **javax.xml.transform.stax** – Streaming API for XML
  - **javax.xml.transform.stream** – I/O stream sources и results

Extensible Stylesheet Language Transformations (XSLT) (2)

- DOM основни класове и интерфейси:
  - TransformerFactory
  - Transformer
  - ErrorListener
  - Result
  - Source
  - SourceLocator
  - Templates
  - URIResolver
  - SAXResult
  - SAXSource
  - DOMResult
  - DOMSource
  - StAXResult
  - StAXSource
  - StreamResult
  - StreamSource

Exercises

...

Streaming API for XML (StAX)

- Java based, event-oriented, Pull Parsing API
- Allows reading as well as writing of XML documents (bi-directional parsing)
- Speed, requires less memory than DOM, easy programming (Iterator desing pattern)
- Pull Parsing several documents can be processed using a single thread
- Two main APIs, which complement each other:
  - Cursor API – fast, using minimal memory footprint, the data is received as primitive types, no garbage
  - Iterator API – data is received as Event objects easier to handle and persist/pass

Streaming API for XML (StAX)

StAX пакети:

- **javax.xml.stream** - StAX programmatic interfaces and object factories allowing reading and writing to/from XML documents using Cursor API and iterator APIs
  - Main classes: XMLEventFactory, XMLInputFactory, XMLOutputFactory
  - Main interfaces: XMLStreamReader, XMLStreamWriter, XMLEventReader, XMLEventWriter
- **javax.xml.stream.events** – StAX programmatic interfaces models the main types of evnets used by the Iterator API
- **javax.xml.stream.util** – additional helper classes for StAX StAX

Streaming API for XML - types of events

- StartDocument
- StartElement
- EndElement
- StartElement.
- Characters
- EntityReference
- ProcessingInstruction
- Comment
- EndDocument
- DTD
- Attribute
- Namespace

StAX Example 1: Reading (1)

```
public class StaxParsingCursor {
public static void main(String[] args) {
   XMLInputFactory inputFactory = null;
   try {
      inputFactory = XMLInputFactory.newFactory();
      inputFactory.setProperty(XMLInputFactory.IS_COALESCING, false);
      inputFactory.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES,
         true);
      inputFactory.setProperty(XMLInputFactory.IS_REPLACING_ENTITY_REFERENCES,
         true);
   } catch (Exception e) {e.printStackTrace();}
   System.out.println("INPUT FACTORY: " + inputFactory);
   try {
      XMLStreamReader streamReader = inputFactory.createXMLStreamReader(
          new FileInputStream("demo.xml"));
```

StAX Example 1: Reading (2)

```
      if (streamReader.getEventType() == XMLStreamReader.START_DOCUMENT) {
         System.out.println("<?xml version=\"" + streamReader.getVersion() +
"\""
          + " encoding=\"" + streamReader.getCharacterEncodingScheme() + "\"?
>");
      }
      while (streamReader.hasNext()) {
         int parsingEventType = streamReader.next();
         if (streamReader.isStartElement()) {
            System.out.print("<" + streamReader.getName());
            int count = streamReader.getAttributeCount();
            for (int i = 0; i < count; i++) {
               System.out.print(" " + streamReader.getAttributeName(i) + "=\""
                              + streamReader.getAttributeValue(i) + "\"");
            }
            System.out.print(">");
         } else if (streamReader.isEndElement()) {
            System.out.print("</" + streamReader.getName() + ">");
```

StAX Example 1: Reading (3)

```
            } else if (streamReader.hasText()) {
                        System.out.print(streamReader.getText());
            }
        }
    } catch (XMLStreamException e) {
        System.out.println(e.getMessage());
            if (e.getNestedException() != null) {
                e.getNestedException().printStackTrace();
             }
        } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

StAX Example 2: Writing (1)

```
public class XHTMLGenarator{
    public static void main(String[] args) throws Exception {
      XMLOutputFactory xof = XMLOutputFactory.newFactory();
      FileOutputStream fos = new FileOutputStream("output.xhtml");
      XMLStreamWriter xsw = xof.createXMLStreamWriter(fos,"utf-8");
      xsw.writeStartDocument("utf-8", "1.0");
      xsw.setPrefix("html", "http://www.w3.org/1999/xhtml");
      xsw.writeComment("XHTML document format");
      xsw.writeStartElement("http://www.w3.org/1999/xhtml", "html");
      xsw.writeNamespace("html", "http://www.w3.org/1999/xhtml");
      xsw.writeStartElement("http://www.w3.org/1999/xhtml", "head");
      xsw.writeStartElement("http://www.w3.org/1999/xhtml",
"title");
      xsw.writeCharacters("RepRap 3D Printer");
      xsw.writeEndElement(); xsw.writeEndElement();
```

```
        xsw.writeStartElement("http://www.w3.org/1999/xhtml", "body");
        xsw.writeStartElement("http://www.w3.org/1999/xhtml", "p");
        xsw.writeCharacters("You can get more information about RepRap
              at ");
        xsw.writeStartElement("http://www.w3.org/1999/xhtml", "a");
        xsw.writeAttribute("href", "http://reprap.org/wiki/RepRap");
        xsw.writeCharacters("this site");
        xsw.writeEndElement(); xsw.writeEndElement();

        xsw.writeEndElement(); xsw.writeEndDocument();
        xsw.writeEndElement();
        xsw.close();
    }
}
```

StAX Example 3: Event API

```
public static void main(String[] args) throws Exception {
    XMLInputFactory factory = XMLInputFactory.newInstance();
    System.out.println("Created factory: " + factory);
    XMLEventReader reader = factory.createXMLEventReader(
          new FileInputStream("myDoc.xml"));
    while (reader.hasNext()) {
        XMLEvent e = reader.nextEvent();
        System.out.println(getEventTypeString(e.getEventType())+ ":" +
e);
    }
}
```

Java™ Architecture for XML Binding (JAXB)

- Дава възможност да се автоматизира процесът на запазване/извличане на данните от/в Java обекти (POJO) към/от XML формат (marshalling/unmarshalling)
- Използва XML Schema като формат за описание на структурата на данните и от него генерира Java класове с JAXB анотации (xjc команда)
- JAXB анотираните класове се използват за автоматично създаване на обекти по време на изпълнение – чрез unmarshaling (десериализация) от XML файл и след това за marshaling (сериализация) обратно от java обектите в XML
- Поддържа се и обратната посока - от JAXB анотираните класове се генерира XML Schema за валидация (schemagen)

Java Architecture for XML Binding (JAXB)

- Java Architecture for XML Binding (JAXB) е стандартна технология в Java SE/EE 5 и нагоре, при която съответствието на XML структури и Java обекти се описва декларативно – с помощта на **анотации** върху **POJO**
- Позволява автоматично генериране на класове от/към XML Schema описание
- Използва се като базова инфраструктура за Java API for XML Web Services (JAX-WS) заедно със StAX
- Лесна за използване и платформено независима
- Позволява генериране на различни от XML сериализации – например JavaScript Object Notation (JSON)

Основни компоненти на JAXB

- XML Schema компилатор: xjc.exe (в bin директорията на Java™ EE сървъра) – генерира JAXB анотирани Java™ класове и пакети:

```
C:\NotesXMLDB>xjc -xmlschema note.xsd -p notification.jaxb -d src
```

- XML Schema генератор: schemagen.exe (в bin директорията на Java™ EE сървъра) – генерира JAXB анотирани Java™ класове и пакети:

```
C:\NotesXMLDB>schemagen -d generated notification.jaxb.Note notification.jaxb.Notes notification.jaxb.ObjectFactory notification\jaxb\package-info.java
```

- Binding Runtime Framework – осъществява unmarshaling /marshaling на java обектите в XML

Пакети на JAXB

- javax.xml.bind – основни класове реализиращи свързване на java обекти и xml елементи за клиентските приложения, включително marshaling (сериализация) / unmarshaling (десериализация) и валидация
- javax.xml.bind.annotation – основни анотации позволяващи свързване (mapping) между java обекти и xml елементи
- javax.xml.bind.annotation.adapters – дефинира адаптерен клас XmlAdapter<ValueType,BoundType>, който позволява да кажем на JAXB как да сериализира / десериализира определени класове с помощта на анотацията @XmlJavaTypeAdapter
- javax.xml.bind.attachment – поддръжка на двоично кодирани прикачени части (attachments)
- javax.xml.bind.helpers – помощни класове (собствен JAXB Provider)

Основни анотации на JAXB (1)

- @XmlAccessorOrder – задава реда на сериализация на JAXB свойствата Java клас или пакет
- @XmlAccessorType – контролира дали се сериализират полета или свойства на Java класа
- @XmlAnyAttribute – обозначава Java поле или свойство от тип java.util.Map, което ще получи всички атрибути, за които не е специфицирана @XmlAttribute анотация
- @XmlAnyElement – обозначава Java поле или свойство от тип списък, което ще получи всички елементи, за които не са специфицирани @XmlElement или @XmlElementRef анотации (съответства на XML схема тип xs:any)
- @XmlAttribute – задава съответствие на поле или свойство на Java класа с атрибут в XML файла

Основни анотации на JAXB (2)

- @XmlElement – задава съответствие на поле или свойство на Java класа с елемент в XML файла (вкл. Namespace, подразб. ст.)
- @XmlElementDecl – свързва XML схема декларация на елемент и метод на ObjectFactory клас, който създава елемента (от тип JAXBElement)
- @XmlElementRef – позволява динамично асоцииране на имена на XML елементи с JavaBean свойства
- @XmlElementRefs – групира @XmlElementRef анотации
- @XmlElements – групира @XmlElement анотации
- @XmlElementWrapper – генерира „обвиващ“ елемент около свойство от тип колекция от стойности (списък, множество, ...)
- @XmlEnum – задава enum клас включващ @XmlEnumValue стойности и задава базовия тип на XML схема типа

Основни анотации на JAXB (3)

- @XmlInlineBinaryData – изключва възможността за оптимизирано двоично кодиране (XOP) на съответното свойство и специфицира че двоичните данни ще бъдат кодирани с base64 encoding
- @XmlList – позволява множество стойности да бъдат специфицирани в общ XML елемент, разделени с интервали
- @XmlMimeType – задава MIME тип на base64-encoded двоични данни (напр. Image или Source)
- @XmlMixed – задава, че съответното свойство съответства на XML елемент с Mixed Content модел
- @XmlNs – специфицира съответствие на пространство от имена и префикс (например на ниво Java пакет)
- @XmlRegistry – анотира клас като фабрика за обекти (ObjectFactory)

Основни анотации на JAXB (4)

- **@XmlRootElement** – често използвана анотация, която декларира коренния елемент на XML документ; може да се приложи към Java™ клас или enumeration
- @XmlSchema – анотира Java™ пакет, като дефинира параметрите на схемата, която пакетът реализира (targetNamespace, elementFormDefault, attributeFormDefault, допълнителни пространства от имена и префикси)
- @XmlSchemaType - свързва клас с прост, вграден XML схема тип
- @XmlTransient -изключва даденото поле от JAXB сериализацията
- @XmlType – дефинира и настройва детайлите на свързването на Java™ клас или enumeration с XML схема тип
- @XmlValue – свързва клас към XML схема сложен тип с просто съдържание (complex type with a simpleContent) или прост тип

JAXB пример: XML схема

```
<complexType>
   <complexContent>
      <restriction base="xs:anyType">
         <sequence>
            <element name="id" type="xs:long"/>
            <element name="to" type="xs:string"/>
            <element name="from" type="xs:string"/>
            <element name="heading" type="xs:string"/>
            <element name="body" type="xs:string"/>
         </sequence>
       </restriction>
    </complexContent>
</complexType>
```

JAXB пример: JAXB клас

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {"id", "to", "from", "heading", "body"})
@XmlRootElement(name = "note")
public class Note {
    protected long id;
    @XmlElement(required = true)
    protected String to;
    @XmlElement(required = true)
    protected String from;
    @XmlElement(required = true)
    protected String heading;
    @XmlElement(required = true)
    protected String body;
...
}
```

Упражнения

...

Референции

- J2EE 1.4 Tutorial – http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
- Java EE 5 Tutorial – http://java.sun.com/javaee/5/docs/tutorial/doc/
- W3Schools – уроци за XML технологии: http://w3schools.com/

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg